**MEMORANDUM**

**DATE:** November 12, 2009

**SUBJECT:** CS565 - Distributed Systems: Project 3

**FROM:** Ryan Middleton, Andy Arminio, Brian Cullinan, Davis Zanot, Talbert Tso

**TO:** Dr. Wolf-Dieter Otte

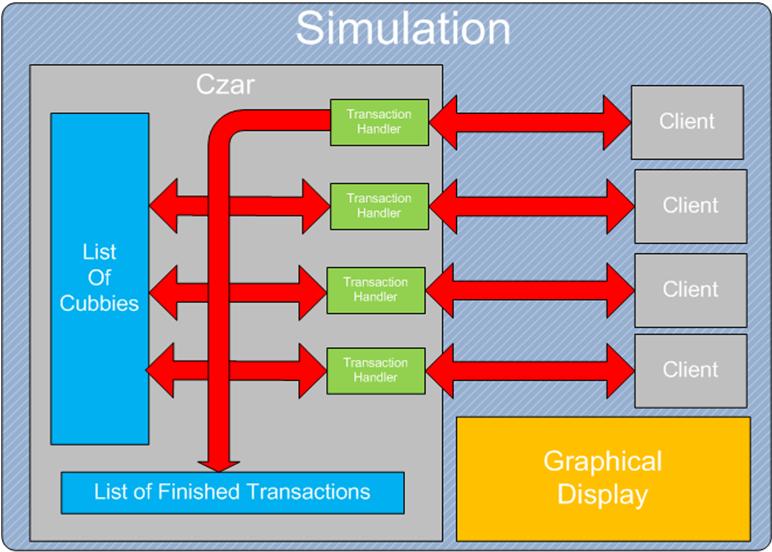
The purpose of this project was to develop a simple transaction control program. Our project uses optimistic concurrency control. When writing this application we made a few assumptions:

* The network is reliable in that eventually transactions will complete
* One transaction per client
* Since each client is a single transaction, it will run as a separate thread

**Abstract Design**

To complete this project we designed a master class that maintains all transactions and “cubby holes”; we named this the czar. We have a client class that connects to the czar and executes a transaction. There are a few steps that take place when executing a transaction:

* A client connects to a transaction handler. In order to achieve and test the concurrency control, each client runs on a separate thread, allowing for multiple clients to be connected to multiple transaction handlers at the same time.
* A client makes a change to the cubby hole using a transaction handler.
* The czar maintains lists of the transaction handlers and the cubby holes. When transaction finish, they are recorded by the czar.
* Then graphical display changes as changes are made to the cubby holes by the transaction handlers.



**Concrete Design**

This project is written in and run on the Java platform. Each class plays a specific role in the execution of transactions.

* The Czar: This is the core component that maintains a connection to a TransactionHandlers. This allows for the TransactionHandler to read and write to the CubbyHoles. It also maintains the list of cubby holes; these are the values that will be changed by the clients. Finally, it records the completed transactions.
* CubbyHole: simply stores an integer value and has methods for reading and writing.
* TransactionHandler: uses the Czar to refer to the cubbies. This contains the logic for overwriting values or reverting back to the original values when there is a conflict. This is the main logic section that controls the transactions.
* Transaction: A Transaction stores the time of execution and when it commits the change to the TransactionHandler it is connected to. It also stores times for when it begins and ends writing. This information is used by the concurrency control when deciding what values are correct.
* Client: The client maintains a connection to a TransactionHandler, It also runs on a separate thread by extending the class Thread. When run, it performs a single Transaction.
* Simulation: The simulation connects the status of the Czar to the Display, this is what is run by Java. This also maintains a list of Clients, just for safety.
* Display: the display reflects the status of the cubby holes, it also provides useful information about the number of conflicts and the initial values.

This class diagram shows the connections between each component.

